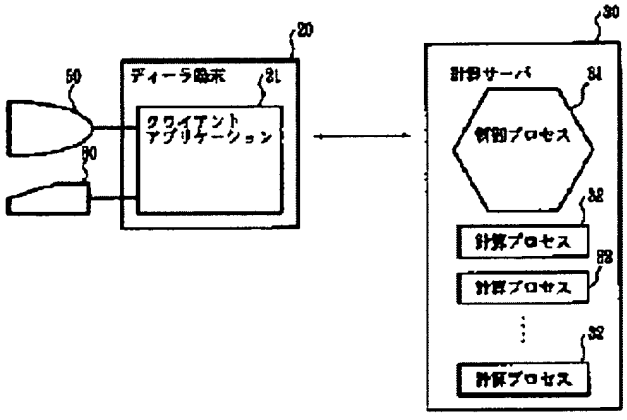
FINANCIAL INFORMATION PROCESSING SYSTEM

Patent number:	JP11259559
Publication date:	1999-09-24
Inventor:	MAEHARA TAKUMI; YAMASHITA YUJI; ITO DAIKI
Applicant:	NIPPON STEEL CORP
Classification:	
- international:	(IPC1-7): G06F17/60
- european:	
Application number:	JP19980061715 19980312
Priority number(s):	JP19980061715 19980312

Abstract of **JP11259559**

PROBLEM TO BE SOLVED: To provide a financial information processing system capable of performing a market-marking processing for plural portfolios in a short time. SOLUTION: A calculation process 32 is for reading one prescribed portfolio from a data base storing the plural portfolios and executing the value washing processing to the one read portfolio. Only one control process 31 is resident on a calculation server 30 at all times. When a market-marking processing request is sent from a dealer terminal 20, the control process 31 decides the number of the simultaneously executable calculation processes 32 and activates the decided number of the calculation processes 32. Then, the activated respective calculation processes 32 are made to parallelly execute the market-marking processing to the plural portfolios corresponding to the market-marking processing request. The result of the market-marking processing is displayed at the display device 50 of the dealer terminal 20.



Family list
1 family member for:
JP11259559
Derived from 1 application.

Back to JP1125

1 FINANCIAL INFORMATION PROCESSING SYSTEM

Inventor: MAEHARA TAKUMI; YAMASHITA YUJI; (+1) **Applicant:** NIPPON STEEL CORP

EC: **IPC:** (IPC1-7): G06F17/60

Publication info: **JP11259559 A** - 1999-09-24

Data supplied from the ***esp@cenet*** database - Worldwide

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平11－259559

(43)公開日　平成11年(1999)９月24日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　17/60		Ｇ０６Ｆ　15/21	Ｑ

審査請求　未請求　請求項の数７　ＯＬ　（全 12 頁）

(21)出願番号　特願平10－61715

(22)出願日　平成10年(1998)３月12日

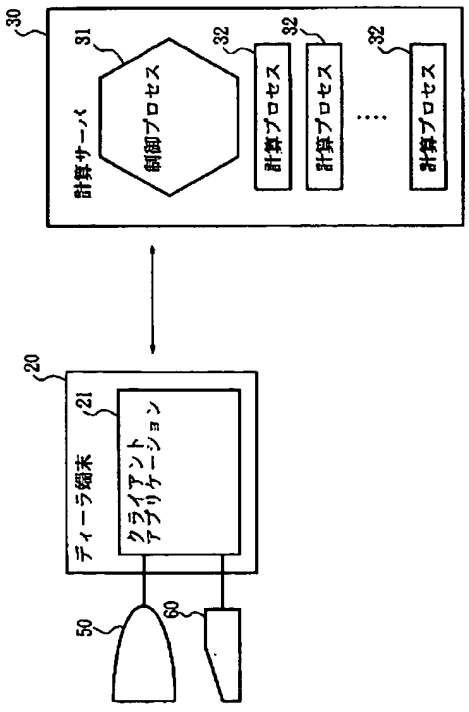
(71)出願人　000006655
新日本製鐵株式会社
東京都千代田区大手町２丁目６番３号
(72)発明者　前原　卓己
東京都千代田区大手町２丁目６番３号　新日本製鐵株式会社内
(72)発明者　山下　雄司
東京都千代田区大手町２丁目６番３号　新日本製鐵株式会社内
(72)発明者　伊藤　大樹
東京都千代田区大手町２丁目６番３号　新日本製鐵株式会社内
(74)代理人　弁理士　半田　昌男

(54)【発明の名称】　金融情報処理システム

(57)【要約】

【課題】　複数のポートフォリオについての値洗処理を短時間で行うことができる金融情報処理システムを提供する。

【解決手段】　計算プロセス３２は、複数のポートフォリオが記憶されているデータベースから所定の一つのポートフォリオを読み出し、その読み出した一つのポートフォリオに対する値洗処理を実行するものである。制御プロセス３１は、計算サーバ３０上に常に一つだけ常駐している。制御プロセス３１は、ディーラ端末２０から値洗処理要求が送られてくると、同時に実行可能な計算プロセス３２の数を決定し、その決定した数だけ計算プロセス３２を立ち上げる。そして、その立ち上げた各計算プロセス３２に、値洗処理要求に対応した複数のポートフォリオについて値洗処理を並列して実行させる。値洗処理の結果は、ディーラ端末２０のディスプレイ装置５０に表示される。



【特許請求の範囲】
【請求項１】　複数の約定データを含む複数のポートフォリオを記憶する記憶手段と、
前記記憶手段から所定の一つ又は複数のポートフォリオを読み出し、その読み出した一つ又は複数のポートフォリオに対する値洗処理を実行する複数の計算プロセス手段と、
一つ又は複数のポートフォリオについて値洗処理を行うべき旨の指示を受けたときに、同時に実行可能な前記計算プロセス手段の数を決定し、その決定した数だけ前記計算プロセス手段を立ち上げ、前記指示された一つ又は複数のポートフォリオについての値洗処理をその立ち上げた前記各計算プロセス手段により並列して実行させる制御プロセス手段と、
前記各計算プロセス手段によって得られた一つ又は複数のポートフォリオについての値洗処理の結果を出力する出力手段と、
を具備し、前記各計算プロセス手段は、一つ又は複数のポートフォリオに対する値洗処理を実行する際に、複数のスレッドを生成し、前記各スレッドにそのポートフォリオに含まれる約定データに対する所定の計算を独立して実行させることを特徴とする金融情報処理システム。
【請求項２】　前記制御プロセス手段は、前記指示を受けたときに、一つの前記計算プロセス手段を立ち上げ、その立ち上げた前記一つの計算プロセス手段に値洗処理を実行させると共に、前記一つの計算プロセス手段による値洗処理の実行中におけるＣＰＵ稼働率の最大値を求めた後、その求めたＣＰＵ稼働率の最大値に基づいて、前記同時に実行可能な前記計算プロセス手段の数を決定することを特徴とする請求項１記載の金融情報処理システム。
【請求項３】　前記制御プロセス手段は、複数のポートフォリオについて前記各計算プロセス手段に値洗処理を並列して実行させているときに、その並列処理の実行中におけるＣＰＵ稼働率が所定の上限値より大きくなったと判断すると、前記同時に実行可能な前記計算プロセス手段の数を減少させ、一方、前記並列処理の実行中におけるＣＰＵ稼働率が所定の下限値よりも小さくなったと判断すると、前記同時に実行可能な前記計算プロセス手段の数を増加させることを特徴とする請求項１記載の金融情報処理システム。
【請求項４】　前記制御プロセス手段は、前記指示を受けたときに、予め設けられた代表的な一つのポートフォリオに対する値洗処理を実行したときのＣＰＵ稼働率の最大値に関するデータに基づいて、前記同時に実行可能な前記計算プロセス手段の数を決定することを特徴とする請求項３記載の金融情報処理システム。
【請求項５】　前記記憶手段は、少なくともＣＰＵ稼働率に関する情報を含むポートフォリオについての処理特性データを記憶しており、前記制御プロセス手段は、前記指示を受けたときに、前記指示に対応したポートフォリオについての前記処理特性データのうち前記記憶手段に記憶されているものを読み出し、前記処理特性データが前記記憶手段に記憶されているポートフォリオについては、前記記憶手段から読み出した前記処理特性データに基づいて、前記同時に実行可能な前記計算プロセス手段の数とポートフォリオの処理順番とを決定し、その決定した数と処理順番とに従って前記各計算プロセス手段に値洗処理を並列して実行させることを特徴とする請求項１記載の金融情報処理システム。
【請求項６】　前記制御プロセス手段は、前記処理特性データが前記記憶手段に記憶されていないポートフォリオについては、一つの前記計算プロセス手段に値洗処理を個別に実行させると共に前記処理特性データを作成して前記記憶手段に記憶させることを特徴とする請求項５記載の金融情報処理システム。
【請求項７】　複数の約定データを含む複数のポートフォリオを記憶する記憶手段から所定の一つ又は複数のポートフォリオを読み出し、その読み出した一つ又は複数のポートフォリオに対する値洗処理を実行する複数の計算プログラムと、
一つ又は複数のポートフォリオについて値洗処理を行うべき旨の指示を受けたときに、同時に実行可能な前記計算プログラムの数を決定し、その決定した数だけ前記計算プログラムを立ち上げ、前記指示された一つ又は複数のポートフォリオについての値洗処理をその立ち上げた前記各計算プログラムにより並列して実行させる制御プログラムと、
を具備し、前記各計算プログラムは、一つ又は複数のポートフォリオに対する値洗処理を実行する際に、複数のスレッドを生成し、前記各スレッドにそのポートフォリオに含まれる約定データに対する所定の計算を独立して実行させることを特徴とするコンピュータ読み取り可能な記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、スワップ、オプションといった金融派生商品（デリバティブ）取引のディーリングやリスク管理業務を支援する金融情報処理システムに関するものである。
【０００２】
【従来の技術】デリバティブ取引では、ディーラは、現在の資産価値はいくらなのか、あるいは資産価値の増減はどの程度なのか等、業務に伴うリスク状態を迅速且つ正確に把握するために、コンピュータを用いて、複数のポートフォリオについて値洗処理を行っている。ここで、ポートフォリオとは、一定の基準で区分された約定データの集まりをいう。また、約定データとは、過去に成立したデリバティブ商品の取引の内容・条件を記録したデータのことである。通常、値洗処理では、金利や為

替レートのような市況データを用いて、一定の計算式によって資産の現在価値を評価している。特に、ポートフォリオ単位で値洗処理を行うことにより、ディーラは、現在のリスク状態をより詳しく評価することができる。

【０００３】

【発明が解決しようとする課題】ところで、金利や為替レート等の市況データは時々刻々と変動しており、これに伴い、現在の資産価値やリスク状態もどんどん変わっていく。このため、ディーラは、実際の市場の動きに連動させて、複数のポートフォリオについて一日に何度でも値洗処理を行い、常に、最新のリスク状態を評価することが望まれる。しかし、値洗処理の対象となるポートフォリオには通常、数十から数千もの約定データが含まれており、また、約定データの数は今後も増大する傾向にある。このため、複数のポートフォリオについての値洗処理には何十分、場合によっては数時間もかかることがある。この処理時間の短縮が課題となっていた。

【０００４】本発明は上記事情に基づいてなされたものであり、複数のポートフォリオについての値洗処理を短時間で行うことができる金融情報処理システムを提供することを目的とするものである。

【０００５】

【課題を解決するための手段】上記の目的を達成するための本発明に係る金融情報処理システムは、複数の約定データを含む複数のポートフォリオを記憶する記憶手段と、前記記憶手段から所定の一つ又は複数のポートフォリオを読み出し、その読み出した一つ又は複数のポートフォリオに対する値洗処理を実行する複数の計算プロセス手段と、一つ又は複数のポートフォリオについて値洗処理を行うべき旨の指示を受けたときに、同時に実行可能な前記計算プロセス手段の数を決定し、その決定した数だけ前記計算プロセス手段を立ち上げ、前記指示された一つ又は複数のポートフォリオについての値洗処理をその立ち上げた前記各計算プロセス手段により並列して実行させる制御プロセス手段と、前記各計算プロセス手段によって得られた一つ又は複数のポートフォリオについての値洗処理の結果を出力する出力手段と、を具備し、前記各計算プロセス手段は、一つ又は複数のポートフォリオに対する値洗処理を実行する際に、複数のスレッドを生成し、前記各スレッドにそのポートフォリオに含まれる約定データに対する所定の計算を独立して実行させることを特徴とするものである。

【０００６】制御プロセス手段は、一つ又は複数のポートフォリオについて値洗処理を行うべき旨の指示を受けると、同時に実行可能な計算プロセス手段の数を決定し、その決定した数だけ計算プロセス手段を立ち上げる。そして、その立ち上げた各計算プロセス手段にマルチプロセスで値洗処理を実行させる。このようにマルチプロセスで値洗処理を行うことにより、本発明では、迅速な値洗処理が可能であり、結果が出るまでの時間を大幅に短縮することができる。

【０００７】また、上記の目的を達成するための本発明に係るコンピュータ読み取り可能な記録媒体は、複数の約定データを含む複数のポートフォリオを記憶する記憶手段から所定の一つ又は複数のポートフォリオを読み出し、その読み出した一つ又は複数のポートフォリオに対する値洗処理を実行する複数の計算プログラムと、一つ又は複数のポートフォリオについて値洗処理を行うべき旨の指示を受けたときに、同時に実行可能な前記計算プログラムの数を決定し、その決定した数だけ前記計算プログラムを立ち上げ、前記指示された一つ又は複数のポートフォリオについての値洗処理をその立ち上げた前記各計算プログラムにより並列して実行させる制御プログラムと、を具備し、前記各計算プログラムは、一つ又は複数のポートフォリオに対する値洗処理を実行する際に、複数のスレッドを生成し、前記各スレッドにそのポートフォリオに含まれる約定データに対する所定の計算を独立して実行させることを特徴とするものである。

【０００８】

【発明の実施の形態】以下に本発明の第一実施形態について図面を参照して説明する。図１は本発明の第一実施形態である金融情報処理システムの全体的な構成を示す図である。図１に示すように、第一実施形態の金融情報処理システムは、ネットワーク１０上に、複数のディーラ端末２０と、ネットワーク１０の中央部に位置する複数の計算サーバ３０と、データベース４０とが接続されている。各ディーラ端末２０は、ディーラが取引業務を行うディーリングルームに設置され、各計算サーバ３０やデータベース４０は、例えば別室に設置されている。両者の間では、プロセス間通信が行われる。各計算サーバ３０はマルチプロセッサ対応の並列コンピュータであり、後述するように、マルチプロセス及びマルチスレッドにより複数のポートフォリオについて値洗処理を実行する。

【０００９】第一実施形態の金融情報処理システムが主として計算の対象とするのは、デリバティブ取引に関連する約定データである。デリバティブ取引とは、原資産の価格に応じ、その価値が決定される商品（デリバティブ商品）を取引するもので、実際の決済は先のことになる。約定データとは、過去に成立したデリバティブ商品の取引の内容・条件を記録したデータのことである。約定データは、金融機関によって異なるが、多い場合には数万から数十万という数に達するが、これらは一定の基準でグループ化されている。各グループ毎に分けられた約定データの集まりがポートフォリオである。グループ分けは、例えば、スワップ、スワップション等の商品毎又は部署毎等により行われる。また、一つのポートフォリオには、通常、数十から数千という約定データが含まれる。

【００１０】ディーラは、今現在保有しているポートフ

ォリオの現在価値や、マーケットの変動により発生し得るリスク等を迅速に評価する必要がある。また、新たに取引の引き合いがあったときに、ディーラは、その引き合いを受けるべきかどうかを短時間に判断するためにも、現在保有する資産価値の時価評価を短時間に知る必要がある。加えて、その資産が、将来のある時点において、どの程度変動している可能性があるかという評価も行う必要がある。

【００１１】しかしながら、約定データに記述されるデリバティブの価値は、実際の金銭の集積ではなく、変動金利、原資産価値に連動して変化する。しかも、一般的に、その評価には高等数学を駆使した経済理論に基づく高度な計算処理が必要で、約定件数が非常に多くなると、約定データを用いた資産の評価は、高性能なコンピュータで行う必要がある。この約定データを使って、現在の資産を評価する計算が、計算サーバ３０によって行われる。かかる現在の資産を評価する計算処理は値洗処理と呼ばれる。特に、第一実施形態では、複数のポートフォリオについてポートフォリオ単位で値洗処理を行うことにする。このようにポートフォリオ単位で値洗処理を行うことにより、ディーラは現在の資産価値を詳しく評価することができる。尚、第一実施形態では、値洗処理の対象となる複数のポートフォリオは、同じような種類のポートフォリオからなるものとする。すなわち、各ポートフォリオについて、約定データの件数、スワップやスワップション等の商品の種類等がほとんど同じであるとする。

【００１２】図２はディーラ端末２０と計算サーバ３０を一台ずつ示している。ディーラ端末２０には、ディスプレイ装置５０及びキーボード６０等が接続されている。ディスプレイ装置５０には、値洗処理の実行状況や処理結果などが表示される。キーボード６０は、ディーラが値洗処理の対象となるポートフォリオを特定するＩＤを指定する場合等に使用される。また、ディーラ端末２０には、値洗処理の結果を紙に出力するプリンタ（不図示）も接続されている。尚、ネットワークを介したプリンタサーバを用いてもよい。

【００１３】ディーラ端末２０の内部では、ネットワーク１０を介しての値洗処理要求の計算サーバ３０への送信、値洗処理の実行状況の受信、値洗処理の結果の受信などを行う金融デリバティブ資産価値評価用アプリケーションソフトウエア（以下単に「クライアントアプリケーション」と称する。）２１が稼働している。ディーラは、ディーリング業務中、複数のポートフォリオについて値洗処理を実行させようと思ったときには、ディーラ端末２０上でその旨の指示を、ポートフォリオのＩＤとともにキーボード６０等から打ち込む。すると、クライアントアプリケーション２１がこれを受け取り、計算サーバ３０に値洗処理要求として送信する。この値洗処理要求には、ポートフォリオを特定するＩＤが含まれている。計算サーバ１２が値洗処理要求を受け取ると、実際に値洗処理を開始する。この具体的に処理内容については後述する。尚、ディーラ端末２０上では、クライアントアプリケーション２１の他にも、デリバティブ取引のリスク管理を行う種々のアプリケーションプログラムが稼働している。

【００１４】図１に示すデータベース４０には、複数のポートフォリオが格納されている。各ポートフォリオに含まれる約定データには、そのポートフォリオのＩＤが付されている。また、現在又は過去の金利や為替レート等の市況データもデータベース４０に格納されている。ディーラ端末２０から計算サーバ３０に値洗処理要求が送られると、計算サーバ３０は、その値洗処理要求と一緒に送られたポートフォリオのＩＤに基づいて、そのＩＤに対応する約定データをデータベース４０から、計算サーバ３０の主メモリ上に読み出す。また、このとき、所定の市況データも主メモリ上に読み出す。

【００１５】計算サーバ３０上では、図２に示すように、制御プロセス３１と計算プロセス３２とが稼働している。計算プロセス３２は、制御プロセス３１による制御のもとで、データベース４０から所定の一つのポートフォリオを読み出し、その読み出した一つのポートフォリオに対する値洗処理を実行するものである。制御プロセス３１は、計算サーバ３０上に常に一つだけ常駐していて、ディーラ端末２０からの値洗処理要求を待っている。制御プロセス３１は、ディーラ端末２０から値洗処理要求が送られてくると、同時に実行可能な計算プロセス３２の数を決定し、その決定した数だけ計算プロセス３２を立ち上げる。そして、その立ち上げた各計算プロセス３２に、値洗処理要求に対応した複数のポートフォリオについて値洗処理を並列して実行させる。すなわち、マルチプロセスにより複数のポートフォリオについての値洗処理が実行される。尚、第一実施形態では、各計算サーバ３０には、複数の計算プロセス３２による値洗処理を実行するＣＰＵを例えば八個搭載している。尚、計算プロセス３２自体を動作させるメインのＣＰＵは別個に設けられていてもよい。

【００１６】また、各計算プロセス３２は、常時、処理状況を制御プロセス３１に通知する。制御プロセス３１は、その処理状況をクライアントアプリケーション２１に送信し、その端末のディスプレイ装置５０に表示させる。図３は計算サーバ３０上における制御プロセス３１と計算プロセス３２との関係を示した図である。図３では、ある一つのディーラ端末２０上で稼働しているクライアントアプリケーション２１が、計算サーバ３０上の制御プロセス３１に対して値洗処理要求を出している状況を示している。この場合、第一実施形態では、制御プロセス３１は、まず、図３（ａ）に示すように、一つの計算プロセス３２だけを立ち上げ、その計算プロセス３２に、値洗処理要求に対応した複数のポートフォリオの

うち、代表的な一つのポートフォリオに対する値洗処理をトライアル的に実行させる。そして、その実行中におけるＣＰＵ稼働率をモニタし、そのＣＰＵ稼働率の最大値を求める。ここで、ＣＰＵ稼働率とは、値洗処理を実行する八個のＣＰＵすべてがフル稼働した場合を１００％としたときに、当該処理の実行中にＣＰＵが何％稼働しているかを表したものをいう。ＣＰＵ稼働率が低いということは、計算サーバ３０の処理能力に余裕があることを意味する。

【００１７】次に、制御プロセス３１は、上記のようにして求めたＣＰＵ稼働率の最大値に基づいて、同時に実行可能な、すなわち、もしいくつかの計算プロセス３２を同時に実行させたとしたときに最大のＣＰＵ稼働率が１００％よりある程度小さくなるような計算プロセス３２の数を決定する。ここで、最大のＣＰＵ稼働率が１００％になるように計算プロセス３２の数を決定しないのは、最大のＣＰＵ稼働率が１００％又はそれ以上では、ＣＰＵに負荷をかけすぎてしまい、かえって処理の効率が悪くなってしまうからである。例えば、トライアル的に実行して求めたＣＰＵ稼働率の最大値が約３０％であったとすると、制御プロセス３１は、同時に実行可能な計算プロセス３２の数を三つに決定する。

【００１８】制御プロセス３１は、この決定した計算プロセス３２の数だけ、実際に計算プロセス３２を立ち上げる。図３（ｂ）では、一例として、制御プロセス３１が三つの計算プロセス３２，３２，３２を立ち上げた場合を示している。そして、制御プロセス３１は、立ち上げた各計算プロセス３２に、トライアル的に実行したポートフォリオ以外の他のポートフォリオについて値洗処理を並列して実行させる。尚、第一実施形態では、値洗処理の対象となる複数のポートフォリオは同じような種類のポートフォリオからなるので、各計算プロセス３２はポートフォリオを任意の順番で処理してよい。

【００１９】次に、各計算プロセス３２によって実行される一つのポートフォリオに対する値洗処理の内容について説明する。一つのポートフォリオに対する値洗処理は、大きく分けて、前処理と、約定計算処理と、後処理という三つの処理からなる。前処理とは、当該計算プロセス３２による処理対象であるポートフォリオをデータベース４０から計算サーバ３０の主メモリ上に読み出し、その読み出したポートフォリオの各約定データに対して、約定計算に入るための準備処理を行うものである。この準備処理では、具体的には、金利データ等を用いて、約定データを後述のスレッドによる計算がしやすいように加工し、その加工した結果の約定データを主メモリに書き込む。

【００２０】尚、各計算プロセス３２がデータベース４０から読み出したポートフォリオはそれぞれ、主メモリ上の異なる領域に一時記憶される。そして、第一実施形態では、マルチアクセスをサポートするためのマルチエンジンを用いて、各計算プロセス３２が主メモリに記憶されたデータの読み出し等を同時に行えるようにしている。

【００２１】約定計算処理は、マルチスレッドで各約定データに対する所定の計算を実行するものである。各計算プロセス３２は、約定計算処理に入ると、まず、スレッドの数を決める。スレッドとは、計算サーバ３０に備えられているＣＰＵの数に応じて分割された処理の単位であり、計算プロセス３２によってソフトウエア的に設定される。すなわち、スレッドの数がその計算プロセス３２が専有するＣＰＵの数となる。スレッドの数が決定されると、計算プロセス３２は、その数だけスレッドを生成し、各スレッドは、ポートフォリオに含まれる各約定データに対する所定の計算を独立して並列に実行する。

【００２２】後処理とは、約定計算処理が終了した以降の処理のことである。実際、約定計算が終わった時点で、ポートフォリオについての最終的な結果が算出されているわけではない。この後処理では、各スレッドで処理された結果を集めて、ポートフォリオについての最終的な結果を計算したり、また、この得られた結果をデータベース４０に書き込んだりする。

【００２３】図４は一つの計算プロセス３２の動作を模式的に示した図である。図４において、入力側の共有メモリ３８は、計算サーバ３０に備えられたＲＡＭ等からなる主メモリである。入力側共有メモリ３８には、この計算プロセス３２が抱えているポートフォリオが記憶され、一つのテーブルとされる。図４の例では、そのポートフォリオに含まれるｎ件の約定データD_1，D_2，・・・，D_nが記憶されている。ポートフォリオは、元々データベース４０に蓄えられており、これを計算プロセス３２が引き出して、入力側の共有メモリ３８に記憶させる。

【００２４】実際に値洗処理を行う場合には、計算プロセス３２は、まず、前処理として、データベース４０から所定のポートフォリオを取ってきて、入力側共有メモリ３８上に書き込む。その後、約定計算処理に入り、ソフトウエア的にスレッドを設定する。図４では、一例として、四つのスレッド３３，３３，３３，３３が設定されている。一つのスレッド３３が、一つのＣＰＵを使って独立に処理を行う単位となるので、四つのスレッド３３，３３，３３，３３がそれぞれ一つずつ、入力側共有メモリ３８から約定データを読み出して、合計四つのＣＰＵを使って並列度４の並列処理を行っている。すなわち、ｎ件の約定データD_1，D_2，・・・，D_nが一つずつ順番に処理されるのではなく、四つのＣＰＵが協調しながらｎ件の約定データD_1，D_2，・・・，D_nを処理する。このようにマルチスレッドで並列的な処理を行うことにより、約定計算処理に要する時間が短縮される。

【００２５】約定データの読み出しの際には、ロック機

構３４の排他制御によって、あるスレッドが入力側共有メモリ３８から約定データを読み出している間は、他のスレッドが約定データの読み出しを行うことができないようになっている。また、同じ約定データを複数のスレッドで処理することがないように、一度あるスレッドによって読み出された約定データは、他のスレッドからは読み出されないようにしている。尚、ロック機構３４は、ＯＳ標準の排他ロック機構によって実現される。この点については後述する。尚、このロック機構３４は、ロックファイルを作ると共に、各スレッドがこのロックファイルを見に行ってアクセスできるかどうかを判断することによって実現するようにしてもよい。

【００２６】各スレッド毎にＣＰＵによって計算されたデータは、出力側の共有メモリ３９に、例えばマトリクス形式のデータとして書き込まれる。この出力側共有メモリ３９も、計算サーバ３０の主メモリである。このとき、各スレッドが無秩序にデータを書き込むと、データ全体が整合がとれないという問題が生じるので、この場合も、ロック機構３５が働いて、あるスレッドが出力側共有メモリ３９にデータを書き込んでいるときには、他のスレッドがデータの書き込みを行うことができないような排他制御を行う。

【００２７】図５は図４に示した入力側の共有メモリ３８上に展開されている約定データのテーブルを示す図である。このテーブルの各行に約定データが置かれており、各行の右端には、１ビットの処理済フラグが設けられている。各スレッドは、自分が処理を実行している約定データには処理済フラグ「１」をセットし、その約定データは既に処理が済んでいることを表示する。他のスレッドは、このテーブルを見るときに、処理済フラグ「１」がセットされている約定データはスキップし、処理済フラグが「０」の約定データを探して読み出す。このとき、第一実施形態では、処理済フラグが「０」の約定データのうち、一番上にある約定データをポインタで指し示すようにしている。スレッドは、このポインタを探して読み出すべきデータを見つけ出す。これにより、スレッドが読み出すべき約定データを簡単に探し出せるようにしている。尚、ポインタを用いずに、スレッドが処理済フラグをスキャンして、「０」のものを見つけ次第それを読み出すようにしてもよい。

【００２８】次に、第一実施形態の金融情報処理システムにおいて、複数のポートフォリオについて値洗処理を行う際の処理の手順について説明する。図６は複数のポートフォリオについて値洗処理を行う際の処理手順を示すフローチャートである。あるディーラが所定のディーラ端末２０から複数のポートフォリオについて値洗処理を行う旨の指示を計算サーバ３０に送ると、計算サーバ３０の制御プロセス３１は、一つの計算プロセス３２を立ち上げて、その計算プロセス３２に特定の一つのポートフォリオについての値洗処理をトライアル的に実行させる（step11）。制御プロセス３１は、そのトライアル的な値洗処理の実行中におけるＣＰＵ稼働率をモニタし、そのＣＰＵ稼働率の最大値を求める。そして、求めたＣＰＵ稼働率の最大値に基づいて、最適な計算プロセス３２の数を決定する（step12）。その後、制御プロセス３１は、step12で決定した数だけ計算プロセス３２を立ち上げ、各計算プロセス３２に複数のポートフォリオについて所定の順番で値洗処理を並列して実行させる（step13）。

【００２９】図７は一つの計算プロセスにおいて行われる処理の手順を示したフローチャートである。制御プロセス３１が、図６のstep13の処理において複数の計算プロセス３２を立ち上げたときに、それぞれの計算プロセス３２において図７に示す処理が開始される。図７の処理手順が開始されると、まず、計算プロセス３２は、制御プロセス３１によって指示された処理対象となるポートフォリオをデータベース４０から計算サーバ３０の主メモリ上に読み出すと共に、スレッドの数を決定し、その決定した数だけスレッドを生成する（step21）。次に、計算プロセス３２は、各スレッドに約定計算をするための演算を並列処理で実行させる（step22）。すなわち、スレッドの数に対応したＣＰＵによって、同時に独立して演算が実行される。並列処理が終了すると、スレッドを消去する（step23）。これにより、一つの計算プロセス３２の処理が終了する。

【００３０】図８は図７に示した処理手順のうちstep22の部分の処理手順を詳しく示したフローチャートである。step30では、まず、スレッドは、図４に示したロック機構３４がロック状態であるかどうか、すなわち別のスレッドがロック機構３４をロック状態にしているかどうかを判断する。ロックされておらず、自分が入力側共有メモリ３８から約定データを読み出せる状態であれば、他のスレッドが読み出しを行うことができないように、ロック機構３４のロックをかける（step31）。次に、スレッドは、約定データの中から、図５に示すようにポインタが指し示している処理済フラグが「０」の約定データを探して、それをフェッチするという処理を行い、そして、その約定データの処理済フラグに「１」をセットする（step32）。このようにスレッドが約定データを読み出して、その約定データの処理済フラグに「１」をセットすると、入力側共有メモリ３８へのアクセスを排除する必要がなくなるので、ロックを解除し（step33）、その後、実際の計算処理に移行する（step34）。計算が終わると、スレッドは、図４に示したロック機構３５がロック状態であるかどうか、すなわち別のスレッドがロック機構３５をロック状態にしているかどうかを判断する（step35）。ロックされておらず、自分が出力側共有メモリ３９に計算結果データを書き込める状態であれば、他のスレッドが書き込みを行うことができないように、ロック機構３５のロックをかける（step

36）。次に、スレッドは、その計算結果データを出力側共有メモリ３９に書き込み（step37）、その後、ロックを解除する（step38）。step39では、未処理の約定データがあるかどうかを判断する。未処理のものがある場合には、上で説明したのと同様の処理を繰り返し、一方、未処理の約定データがなくなった場合には、処理を終了する。

【００３１】次に、第一実施形態の金融情報処理システムを用いて、複数のポートフォリオについての値洗処理を、上述したマルチプロセス及びマルチスレッドによる処理方式（本発明の処理方式）で行った場合と、マルチプロセスによらずマルチスレッドだけによる処理方式（マルチスレッド処理方式）で行った場合と、マルチプロセスやマルチスレッドを用いない従来の処理方式で行った場合とについて、比較して説明する。図９（ａ）は従来の処理方式で複数のポートフォリオについての値洗処理を行ったときのＣＰＵ稼働率と処理時間との関係を示す図、図９（ｂ）はマルチスレッド処理方式で複数のポートフォリオについての値洗処理を行ったときのＣＰＵ稼働率と処理時間との関係を示す図、図９（ｃ）は本発明の処理方式で複数のポートフォリオについての値洗処理を行ったときのＣＰＵ稼働率と処理時間との関係を示す図である。ここで、図９（ａ），（ｂ）及び（ｃ）において、縦軸はＣＰＵ稼働率を、横軸は処理時間を表す。

【００３２】従来の処理方式では、一つのＣＰＵが各ポートフォリオについて一つずつシーケンシャルに処理すると共に、そのホートフォリオに含まれる各約定データについても一つずつ処理することになる。第一実施形態では、ＣＰＵ稼働率は、八個のＣＰＵすべてがフルに稼働したときを１００％として定められる。従来の処理方式では、常時、一つのＣＰＵしか稼働していないため、当然、ＣＰＵ稼働率は、図９（ａ）に示すように、各ポートフォリオについて前処理、約定計算処理及び後処理を行う全期間にわたってかなり小さい。したがって、すべてのポートフォリオについて値洗処理が終了するまでの処理時間は非常に長い。

【００３３】一方、マルチスレッド処理方式では、各ポートフォリオについて一つずつシーケンシャルに値洗処理を行うが、その値洗処理の中で約定計算処理についてはマルチスレッドで処理することになる。このため、図９（ｂ）に示すように、各ポートフォリオについて約定計算処理を行う期間では、ＣＰＵ稼働率が高くなる。これに対し、前処理又は後処理を行う期間では、従来の処理方式でもマルチスレッド処理方式でも、ＣＰＵ稼働率は同じである。また、一つのポートフォリオについて約定計算処理を行うときの処理量は、図９（ａ）又は（ｂ）において、その約定計算処理を行っている期間に対応する、ＣＰＵ稼働率を表す曲線と横軸とで囲まれた面積で表される。かかる処理量は、当然のことながら、従来の処理方式を用いた場合であろうと、マルチスレッド処理方式を用いた場合であろうと変わらない。このため、マルチスレッド処理方式により値洗処理を行う場合は、約定計算処理を行う期間におけるＣＰＵ稼働率が高いので、従来の処理方式により値洗処理を行う場合に比べて、約定計算処理に要する処理時間が短くなる。したがって、すべてのポートフォリオについて値洗処理が終了するまでの処理時間が短縮する。

【００３４】ところで、現在、一つのポートフォリオには数十から数千の約定データが含まれている。かかる程度の規模の約定データを含むポートフォリオを、第一実施形態の金融情報処理システムを用いてマルチスレッド処理方式で値洗処理を行う場合、通常、約定計算処理の期間におけるＣＰＵ稼働率は例えば約２０～４０％であったとすると、ＣＰＵの処理能力にまだ余裕がある。本発明の処理方式は、この点に着目し、余裕があるＣＰＵの処理能力を有効に活用して、複数の計算プロセス３２に複数のポートフォリオについて値洗処理を並列して実行させるものである。したがって、ポートフォリオについての値洗処理はいくつか同時に実行されるので、ＣＰＵ稼働率は、図９（ｃ）に示すように、極めて高くなる。このため、複数のポートフォリオについての値洗処理が終了するまでの処理時間は、マルチスレッド処理方式により値洗処理を行う場合に比べて、大幅に短縮する。

【００３５】第一実施形態の金融情報処理システムでは、複数の計算プロセスに複数のポートフォリオについての値洗処理を並列して実行させることにより、迅速な処理が可能であり、結果が出るまでの時間を大幅に短縮することができる。このため、ディーラは、実際の市場の動きに連動させて、複数のポートフォリオについて一日に何度でも値洗処理を行い、常に、最新のリスク状態を迅速に評価することが可能となる。

【００３６】次に、本発明の第二実施形態である金融情報処理システムについて説明する。第二実施形態の金融情報処理システムでは、同時に実行可能な計算プロセスの数を決定する方法が、上記第一実施形態の方法と異なる。また、上記の第一実施形態では、値洗処理の対象となる複数のポートフォリオが同じような種類のポートフォリオの集まりである場合を考えていた。しかし、第二実施形態では、値洗処理の対象となる複数のポートフォリオには、種類の異なるポートフォリオが含まれていてもよい。尚、第二実施形態において第一実施形態のものと同一の機能を有するものには、同一の符号を付すことにより、その詳細な説明を省略する。

【００３７】第二実施形態では、計算サーバ３０の制御プロセス３１は、予め代表的な一つのポートフォリオに対する値洗処理を実行させたときのＣＰＵ稼働率の最大値に関するデータを持っている。制御プロセス３１は、ディーラ端末２０から複数のポートフォリオについて値

洗処理を実行する旨の指示を受けたときに、ＣＰＵ稼働率の最大値に関するデータに基づいて、同時に実行可能な計算プロセス３２の数の初期値を決定する。そして、その決定した初期値だけ計算プロセス３２を立ち上げて、マルチプロセスで値洗処理を実行させる。

【００３８】また、制御プロセス３１は、ＣＰＵ稼働率に関する下限値T_a及び上限値T_bのデータを予め持っている。制御プロセス３１は、複数の計算プロセス３２に値洗処理を実行させている間、常時、ＣＰＵ稼働率をモニタしている。そして、ＣＰＵ稼働率が範囲［T_a，T_b］に含まれているときは、制御プロセス３１は、そのまま複数の計算プロセス３２に処理を実行させる。一方、制御プロセス３１は、ＣＰＵ稼働率が下限値T_aより小さくなったと判断すると、同時に実行させるべき計算プロセスの数を一つ増加することを決定する。また、ＣＰＵ稼働率が上限値T_bより大きくなったと判断すると、同時に実行させるべき計算プロセス３２の数を一つ減少することを決定する。そして、かかる決定をすると、制御プロセス３１は、実際にその決定した数の計算プロセス３２に値洗処理を並列して実行させる。このように、第二実施形態では、ＣＰＵ稼働率が一定の範囲内に収まるように、制御プロセス３１が同時実行させる計算プロセス３２の数を自動的に決定することにより、値洗処理の対象となる複数のポートフォリオに、種類の異なるポートフォリオが含まれていても、値洗処理を効率よく実行させることができる。

【００３９】図１０は第二実施形態の金融情報処理システムにおいて複数のポートフォリオについて値洗処理を行う際の処理手順を示すフローチャートである。あるディーラがディーラ端末２０から複数のポートフォリオについて値洗処理を行う旨の指示を計算サーバ３０に送ると、制御プロセス３１は、予め設けられたＣＰＵ稼働率の最大値に関するデータに基づいて、同時に実行可能な計算プロセス３２の数を決定する（step41）。次に、制御プロセス３１は、決定した数だけ計算プロセス３２を立ち上げ、各計算プロセス３２に値洗処理を並列して実行させる（step42）。そして、値洗処理を実行している間（step43）、制御プロセス３１は、常時、ＣＰＵ稼働率をモニタして、ＣＰＵ稼働率が所定の範囲［T_a，T_b］内に収まっているか否かを判断する（step44）。ＣＰＵ稼働率が所定の範囲［T_a，T_b］内に収まっていれば、そのまま各計算プロセス３２に値洗処理を実行させる（step42）。一方、ＣＰＵ稼働率が所定の範囲［T_a，T_b］内に収まっていないと判断すると、制御プロセス３１は、同時に実行させるべき計算プロセス３２の数を変更することを決定し（step45）、その変更した数の計算プロセス３２に値洗処理を実行させる（step42）。そして、制御プロセス３１が処理対象のすべてのポートフォリオについて値洗処理が終了したと判断すると（step43）、図１０のフローから抜ける。

【００４０】第二実施形態の金融情報処理システムでは、上記第一実施形態のものと同様に、複数の計算プロセスに複数のポートフォリオについての値洗処理を並列して実行させることにより、迅速な処理が可能であり、結果が出るまでの時間を大幅に短縮することができる。また、第二実施形態の金融情報処理システムでは、マルチプロセスで値洗処理を実行させている間に、制御プロセスが、ＣＰＵ稼働率が所定の範囲内に収まっているか否かを判断し、それに応じて、計算プロセスの数を変更することにより、値洗処理の対象となる複数のポートフォリオに種類の異なるポートフォリオが含まれている場合でも、値洗処理の効率を向上させることができるので、より迅速な処理が可能である。

【００４１】次に、本発明の第三実施形態である金融情報処理システムについて説明する。第三実施形態の金融情報処理システムでは、同時に実行可能な計算プロセスの数を決定する方法が、上記第一及び第二実施形態の方法と異なる。また、上記の第一実施形態では、値洗処理の対象となる複数のポートフォリオが同じような種類のポートフォリオの集まりである場合を考えていた。しかし、第三実施形態では、第二実施形態と同様に、値洗処理の対象となる複数のポートフォリオには、種類の異なるポートフォリオが含まれていてもよい。尚、第三実施形態において第一実施形態のものと同一の機能を有するものには、同一の符号を付すことにより、その詳細な説明を省略する。

【００４２】第三実施形態の金融情報処理システムでは、データベース４０は、複数のポートフォリオを格納すると共に、ポートフォリオに対する値洗処理の特性を記した処理特性データを表形式で格納している。図１１はポートフォリオの処理特性データの集まりである処理特性テーブルを説明するための図である。かかる処理特性テーブルには、図１１に示すように、ポートフォリオのＩＤと、ポートフォリオの種類と、ポートフォリオに対する値洗処理の実行中におけるＣＰＵ稼働率の最大値と、ポートフォリオに対する値洗処理に要する処理時間と、ポートフォリオに含まれる約定データの件数とが記載される。ここで、ポートフォリオの種類の欄には、例えば、このポートフォリオに含まれる約定データが、スワップだけであるとか、スワップとスワップションとからなるとかが記載される。

【００４３】尚、ポートフォリオの処理特性データを、予めすべてのポートフォリオに対して作成し、データベース４０に記憶させておく必要はない。第三実施形態では、処理特性データが作成されていないポートフォリオについては、後述するように、値洗処理要求が出され、実際に値洗処理を実行する際に、処理特性データが作成される。

【００４４】計算サーバ３０の制御プロセス３１は、ディーラ端末２０から複数のポートフォリオについて値洗

処理を実行する旨の指示を受けたときに、その指示に対応したポートフォリオの処理特性データのうち、データベース４０に記憶されているものを、計算サーバ３０の主メモリ上に読み出す。そして、制御プロセス３１は、処理特性データがデータベース４０に記憶されていないポートフォリオについては、一つの計算プロセス３２に値洗処理を個別に実行させると共に、処理特性データを作成してデータベース４０に記憶させる。一方、処理特性データがデータベース４０に記憶されているポートフォリオについては、制御プロセス３１は、データベース４０から読み出した処理特性データに基づいて、同時に実行可能な計算プロセス３２の数と、ポートフォリオの処理順番とを決定する。そして、その決定した数だけ計算プロセス３２を立ち上げ、それらの計算プロセス３２に、その決定した処理順番に従って値洗処理を並列して実行させる。

【００４５】次に、計算プロセス３２の数とポートフォリオの処理順番との決定の仕方を具体的に説明する。例えば、処理対象となる複数のポートフォリオのうち、二つのポートフォリオがスワップションを多数含むものであり、他のポートフォリオがスワップのみを含むものであるとする。ここで、スワップションを多数含むポートフォリオは、ＣＰＵ稼働率の最大値が例えば４０～５０％であり、スワップのみを含むポートフォリオは、ＣＰＵ稼働率の最大値が例えば２０～３０％である。この場合、制御プロセス３１は、最初に、二つの計算プロセス３２を立ち上げ、その二つの計算プロセス３２に、スワップションを多数含むポートフォリオについての値洗処理を同時に実行させる。次に、かかるスワップションを多数含む二つのポートフォリオについての値洗処理が終了した後、もう一つの計算プロセス３２を立ち上げ、合計三つの計算プロセス３２に、スワップのみを含むポートフォリオについての値洗処理を同時に実行させる。このように、制御プロセス３１は、複数のポートフォリオについての値洗処理が効率よく行われるように、計算プロセス３２の数とポートフォリオの処理順番を決定する。

【００４６】図１２は第三実施形態の金融情報処理システムにおいて複数のポートフォリオについて値洗処理を行う際の処理手順を示すフローチャートである。あるディーラがディーラ端末２０から複数のポートフォリオについて値洗処理を行う旨の指示を計算サーバ３０に送ると、制御プロセス３１は、その指示に対応したポートフォリオについての処理特性データをデータベース４０から主メモリ上に読み出し、処理対象のポートフォリオのすべてについて処理特性データがデータベース４０に記憶されているか否かを判断する（step51）。処理対象のすべてのポートフォリオについて処理特性データがあると判断すると、step54に移行する。一方、処理特性データのないポートフォリオが少なくとも一つあると判断すると、制御プロセス３１は、一つの計算プログラム３２を立ち上げ、その計算プログラム３２に、その処理特性データがないポートフォリオについて個別に値洗処理を実行させる（step52）。そして、制御プロセス３１は、かかる処理の際に、ＣＰＵ稼働率や処理時間等をモニタし、そのポートフォリオについての処理特性データを新たに作成して、これをデータベース４０に格納する（step53）。その後、step54に移行する。尚、処理対象のポートフォリオのすべてについて処理特性データがない場合には、step52の値洗処理によって結果がすべて得られているので、step54以下のステップに進むことなく、このstep53で処理が終了する。

【００４７】次に、step54では、制御プロセス３１は、step51の処理の際に読み出した処理特性データに基づいて、同時に実行可能な計算プロセス３２の数を決定すると共に、ポートフォリオの処理順番を例えば稼働率の大きい順に決定する（step54）。そして、制御プロセス３１は、決定した数だけ計算プロセス３２を立ち上げ、各計算プロセス３２に、決定した処理順番に従って、step51で読み出した処理特性データに対応するポートフォリオについて値洗処理を並列して実行させる（step55）。

【００４８】このように、新たなポートフォリオについては、処理特性データを作成するために、値洗処理が個別に行われ、一方、すでに処理特性データが作成されているポートフォリオについては、マルチプロセスにより値洗処理が行われる。したがって、今回の処理対象となるポートフォリオのすべてが、これまでに少なくとも一回値洗処理が行われたポートフォリオであれば、今回の値洗処理では、step52及びstep53のステップを全く経由せずに、step54のステップから処理が行われることになるので、処理時間が大幅に短縮する。

【００４９】尚、ポートフォリオは常に同じであるわけではなく、将来、それに含まれる約定データの件数が増加したり、それに伴って種類が変化することがある。かかる場合に対処するために、処理特性データを一定の期間毎に更新するようにしてもよい。第三実施形態の金融情報処理システムでは、上記第一実施形態のものと同様に、複数の計算プロセスに複数のポートフォリオについての値洗処理を並列して実行させることにより、迅速な処理が可能であり、結果が出るまでの時間を大幅に短縮することができる。また、第三実施形態の金融情報処理システムでは、ポートフォリオの処理特性データに基づいて、同時に実行可能な計算プロセスの数とポートフォリオの処理順番とを決定することにより、値洗処理の対象となる複数のポートフォリオに種類の異なるポートフォリオが含まれている場合でも、値洗処理の効率を向上させることができる。

【００５０】尚、本発明は上記の各実施形態に限定されるものではなく、その要旨の範囲内において種々の変形が可能である。上記の各実施形態では、約定計算処理を

マルチスレッドで行う場合について説明したが、約定計算処理を必ずしもマルチスレッドで行う必要はない。但し、この場合は、当然のことながら、マルチプロセスとマルチスレッドで値洗処理を行う場合に比べて処理時間が遅くなる。

【００５１】また、上記の各実施形態では、各計算サーバに、値洗処理を実行するＣＰＵを八個搭載した場合について説明したが、一般に、八個以外の任意の数のＣＰＵを各計算サーバに搭載してもよい。極端な場合、各計算サーバは一つのＣＰＵを用いたものであってもよい。

【００５２】

【発明の効果】以上説明したように本発明に係る金融情報処理システムによれば、複数の計算プロセスに複数のポートフォリオについての値洗処理を並列して実行させることにより、迅速な処理が可能であり、結果が出るまでの時間を大幅に短縮することができる。このため、ディーラは、実際の市場の動きに連動させて、複数のポートフォリオについて一日に何度でも値洗処理を行い、常に、最新のリスク状態を迅速に評価することが可能となる。

【図面の簡単な説明】

【図１】本発明の第一実施形態である金融情報処理システムの全体的な構成を示す図である。

【図２】その金融情報処理システムのうちディーラ端末と計算サーバを一台ずつ示した図である。

【図３】計算サーバ上における制御プロセスと計算プロセスとの関係を示した図である。

【図４】計算プロセスの動作を模式的に示した図である。

【図５】図４に示す入力側共有メモリ上に展開されている約定データのテーブルを示す図である。

【図６】第一実施形態の金融情報処理システムにおいて複数のポートフォリオについて値洗処理を行う際の処理手順を示すフローチャートである。

【図７】一つの計算プロセスにおいて行われる処理の手順を示したフローチャートである。

【図８】図７に示した処理手順の一部の処理手順を詳しく示したフローチャートである。

【図９】従来の処理方式、マルチスレッド処理方式及び本発明の処理方式のそれぞれで複数のポートフォリオについての値洗処理を行ったときのＣＰＵ稼働率と処理時間との関係を示す図である。

【図１０】第二実施形態の金融情報処理システムにおいて複数のポートフォリオについて値洗処理を行う際の処理手順を示すフローチャートである。

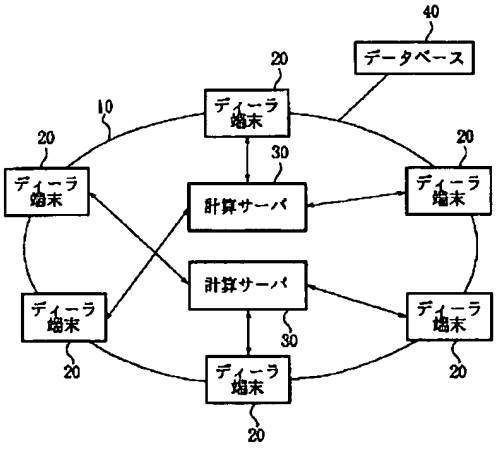
【図１１】ポートフォリオの処理特性データの集まりである処理特性テーブルを説明するための図である。

【図１２】第三実施形態の金融情報処理システムにおいて複数のポートフォリオについて値洗処理を行う際の処理手順を示すフローチャートである。

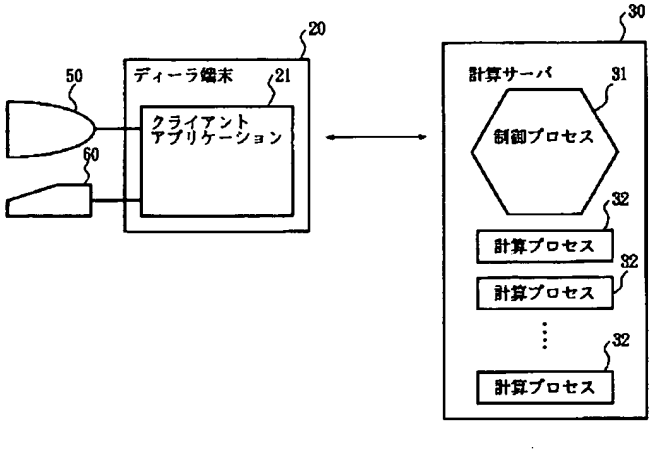
【符号の説明】

１０　ネットワーク
２０　ディーラ端末
２１　クライアントアプリケーション
３０　計算サーバ
３１　制御プロセス
３２　計算プロセス
３３　スレッド
３４，３５　ロック機構
３８　入力側共有メモリ
３９　出力側共有メモリ
４０　データベース
５０　ディスプレイ装置
６０　キーボード

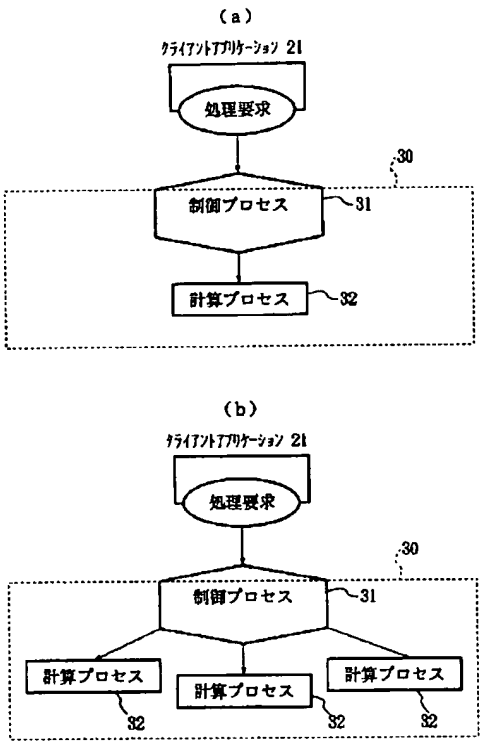
【図１】



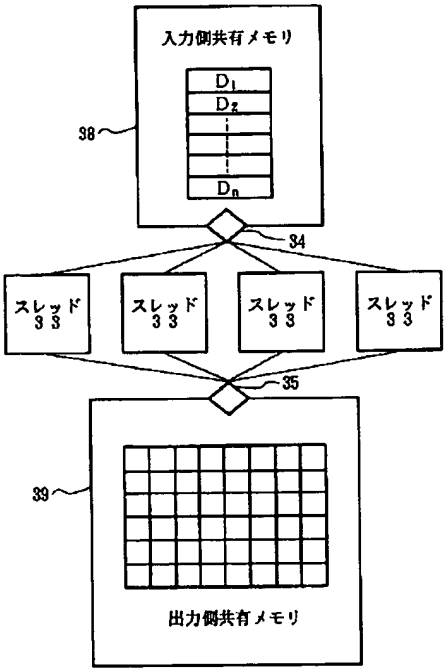
【図２】



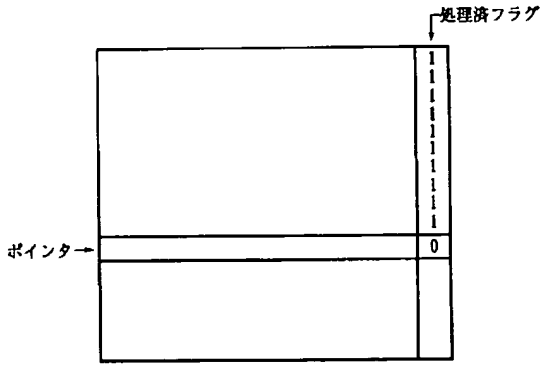
【図３】



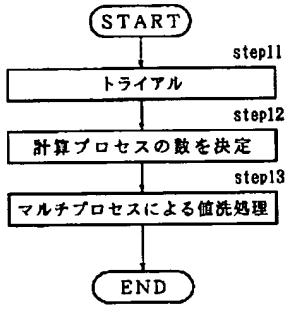
【図４】



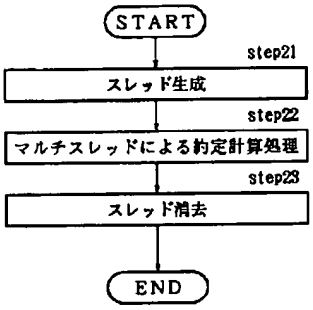
【図５】



【図６】



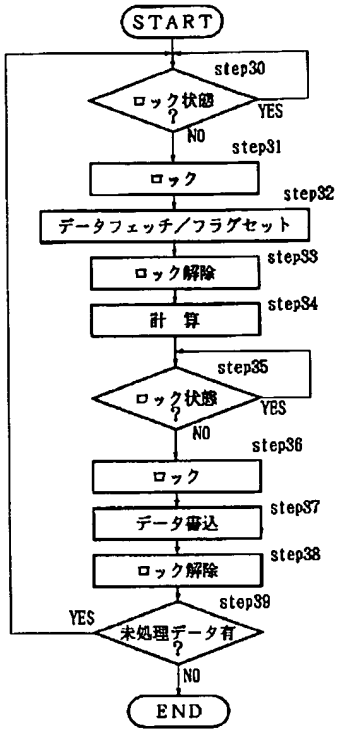
【図７】



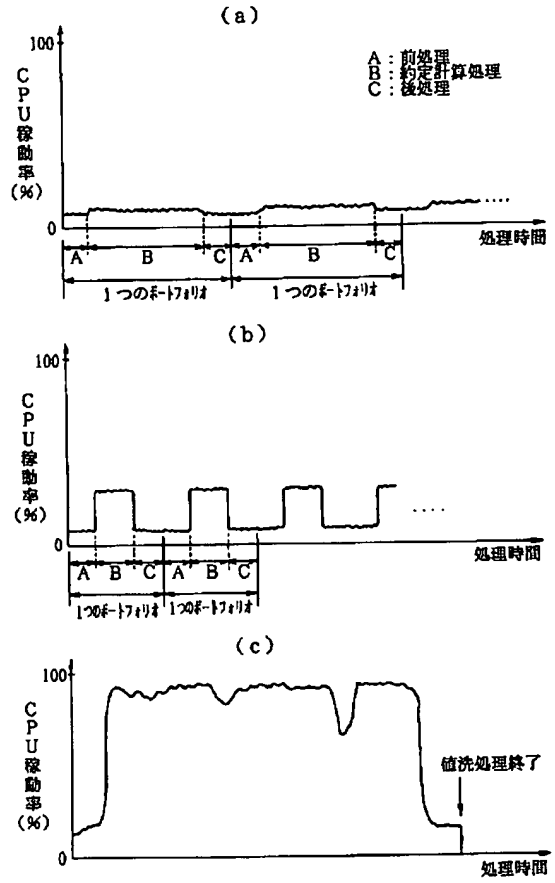
【図１１】

ポートフォリオＩＤ	種類	ＣＰＵ稼動率の最大値	処理時間	約定件数
⋮	⋮	⋮	⋮	⋮

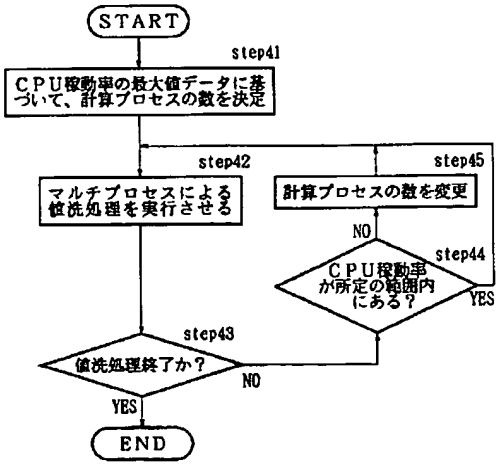
【図８】


START
step30
ロック状態？
YES
NO
step31
ロック
step32
データフェッチ／フラグセット
step33
ロック解除
step34
計　算
step35
ロック状態？
YES
NO
step36
ロック
step37
データ書込
step38
ロック解除
step39
YES
未処理データ有？
NO
END

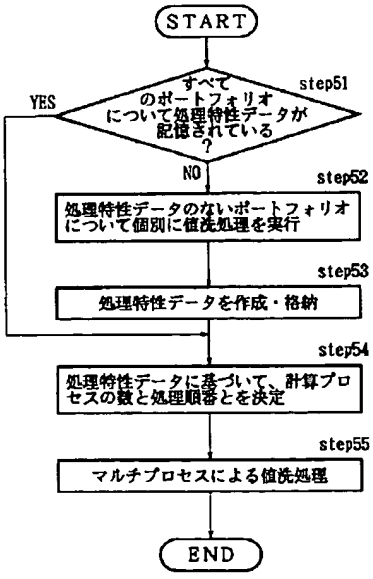
【図９】


(a)
100
CPU稼動率(%)
0
A：前処理
B：約定計算処理
C：後処理
処理時間
A
B
C
A
B
C
1つのポートフォリオ
1つのポートフォリオ
(b)
100
CPU稼動率(%)
0
処理時間
A
B
C
A
B
C
1つのポートフォリオ
1つのポートフォリオ
(c)
100
CPU稼動率(%)
0
値洗処理終了
処理時間

【図１０】


START
step41
CPU稼動率の最大値データに基づいて、計算プロセスの数を決定
step42
マルチプロセスによる値洗処理を実行させる
step45
計算プロセスの数を変更
NO
step44
CPU稼動率が所定の範囲内にある？
YES
step43
値洗処理終了か？
NO
YES
END

【図１２】


START
YES
すべてのポートフォリオについて処理特性データが記憶されている？
step51
NO
step52
処理特性データのないポートフォリオについて個別に値洗処理を実行
step53
処理特性データを作成・格納
step54
処理特性データに基づいて、計算プロセスの数と処理順番とを決定
step55
マルチプロセスによる値洗処理
END